

Learning Stochastic Deterministic Regular Languages

Franck Thollard and Alexander Clark

EURISE,
Jean Monnet University, Saint-Etienne
`franck.thollard@univ-st-etienne.fr`

ISSCO/TIM, Université de Genève
40 Bvd du Pont d'Arve CH-1211, Genève 4, Switzerland
`asc@aclark.demon.co.uk`

Abstract. We propose in this article a new practical algorithm for inferring μ -distinguishable stochastic deterministic regular languages. We prove that this algorithm will infer, with high probability, an automaton isomorphic to the target when given a polynomial number of examples. We discuss the links between the error function used to evaluate the inferred model and the learnability of the model class in a PAC like framework.

1 Introduction

Probabilistic automata are compact and efficient devices. They are used in a number of domains including natural language processing [21, 25], speech processing [19, 23, 24] and information retrieval [14, 27].

The increasing amount of data available in different research fields allows the automatic building of these devices. The learnability of Stochastic Deterministic Regular Languages (SDRL) has been studied in the identification in the limit framework. [6] showed that the structure of the target automaton could be identified in the limit with probability one while [9] showed that rational probabilities could be exactly identified. However, the use of these algorithms is still problematic in practice since it is never known if the amount of data available is enough to achieve the identification. For these reasons, we are interested here in a PAC like framework. A subclass of the class of SDRLs has already been studied [22] and the goal here will be the extension of this work to the class of all SDRLs.

Note that some authors have studied machines that define probability distributions over Σ^n or over $\Sigma^{\leq n}$ for a given n [1, 15, 22]. Even if n can be chosen arbitrarily large, these models still define probability distributions over a *finite* set of strings. We study here the inference of probability distributions over Σ^* .

The paper is organized as follow. We first present a brief overview of probabilistic grammatical inference and present our new algorithm. We then show

that, if enough data is available, the algorithm will infer, with high probability, an automaton isomorphic to the target. We then discuss the link between the error function used to estimate the quality of the inferred model and the learnability of the model class.

2 Preliminaries

An alphabet Σ is a finite set of symbols. We will denote by Σ^* the free monoid. Symbols from Σ will be denoted by $\sigma_1, \sigma_2, \dots$ and strings from Σ^* by end of alphabet letters (*e.g.* x, y, z).

A probability distribution over Σ^* (written \mathcal{D}) is a function $\Sigma^* \rightarrow [0, 1]$ such that $\sum_{x \in \Sigma^*} \Pr_{\mathcal{D}}(x) = 1$. A distribution modeled by a machine (*e.g.* a probabilistic automaton) A , will be written \mathcal{D}_A , or A in a non ambiguous context. We will use the L_∞ distance in order to measure the dissimilarity (or similarity) between two distributions: $L_\infty(\mathcal{D}_1, \mathcal{D}_2) = \max_{x \in \Sigma^*} |\Pr_{\mathcal{D}_1}(x) - \Pr_{\mathcal{D}_2}(x)|$.

The goal here will be the inference of stochastic deterministic regular languages. Even if more general model classes could be inferred (see for example the heuristic techniques of [4] for the learning of Hidden Markov Models or [13] for the learning of residual automata) we prefer the deterministic version for its efficiency at parsing time.

Definition 1. A DPFA is a tuple $A = \langle Q_A, \Sigma, \delta_A, q_0, F_A, P_A \rangle$, where:

- Q_A is a finite set of states;
- Σ is an alphabet;
- $\delta_A \subseteq Q_A \times \Sigma \times Q_A$ is a set of transitions;
- $q_0 \in Q_A$ is the initial state;
- $P_A : \delta_A \rightarrow (0, 1]$ is the probability transition function;
- $F_A : Q_A \rightarrow [0, 1]$ is the stopping probability function.

For the automaton to be deterministic, δ_A must be a function graph, that is

$$\forall q \in Q_A, \forall \sigma \in \Sigma, |\{q' \in Q_A : (q, \sigma, q') \in \delta_A\}| \leq 1 \quad (1)$$

Moreover, the automaton must satisfy the following conditions in order to define a probability distribution: For each state $q \in Q_A$,

$$F_A(q) + \sum_{\sigma \in \Sigma, q' \in Q_A} P_A(q, \sigma, q') = 1. \quad (2)$$

Moreover, all accessible states must recognize at least one string of strictly positive probability.

For a given transition $\tau = (q_i, \sigma, q_j)$, state q_i (resp. q_j) will be the origin (resp. the end) of τ . Using deterministic automata, a transition (q_i, σ, q_j) is totally defined by q_i and σ and thus can be equivalently written (q_i, σ, \bullet) .

A *path* π in a DPFA A is a sequence of transitions $\pi = \tau_1\tau_2\dots\tau_n$ with $\tau_i \in \delta_A$. Abusing notation, we will say that the origin (resp. the end) of the path is the origin of its first (resp. its last) transition. We will note $A \overset{x}{\rightsquigarrow} q_n$ the path in A that recognizes the string $x = \sigma_1\sigma_2\dots\sigma_n$ and has q_n as the end of the path; in other words, $A \overset{x}{\rightsquigarrow} q_n$ denotes the sequence of transitions $\pi_A(x) = (q_0, \sigma_1, q_1) \dots (q_{n-1}, \sigma_n, q_n)$ in A .

The probability of a path π will be written $\Pr_A(\pi)$ and computed as follow: $\Pr_A(\pi) = \prod_{\tau_i \in \pi} P_A(\tau_i)$. The probability of a string $x = \sigma_1\sigma_2\dots\sigma_n$ according to the DPFA A is computed through:

$$\Pr_{\mathcal{D}_A}(x) = \Pr_A(A \overset{x}{\rightsquigarrow} q_n) \bullet F_A(q_n) \quad (3)$$

As an example, for $x = ab$, and the DPFA A of figure 1, the path $\pi_A(ab) = (0, a, 1)(1, b, 3)$ has probability $\Pr_A(\pi_A(ab)) = 0.125 \times 0.4$. The probability of the string ab is $\Pr_A(ab) = \Pr_A(\pi_A(ab))F_A(3) = 0.125 \times 0.4 \times 1$.

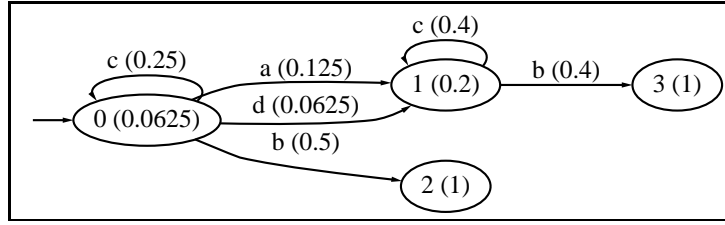


Fig. 1. A DPFA

As soon as a DPFA A satisfies the conditions (2) it defines a probability distribution over Σ^* , that is $\sum_{x \in \Sigma^*} \Pr_{\mathcal{D}_A}(x) = 1$.

We will note $A_{[q_0=q_j]}$ the DPFA A in which the initial state has been set to q_j . Note that $A_{[q_0=q_j]}$ is still a DPFA in that it respects conditions 1 and 2.

Definition 2 (μ -distinguishability). A DPFA A is μ -distinguishable if, for each pair of states q_i, q_j , the distance between distributions $\mathcal{D}_{A_{[q_0=q_i]}}$ and $\mathcal{D}_{A_{[q_0=q_j]}}$ is greater than μ : $\forall q_i, q_j \in Q_A, L_\infty(\mathcal{D}_{A_{[q_0=q_i]}}, \mathcal{D}_{A_{[q_0=q_j]}}) > \mu$.

The set of distributions modeled by μ -distinguishable DPFA will be denoted by μ -SDRL. Unless specified, from now, distributions will be a shortcut for μ -SDRL over Σ^* .

3 Learning distributions

While learning a concept, the learner (or the learning algorithm) is given a certain amount of information related to the target concept. In our case, the aim is the approximation of μ -SDRL. Let $\mathcal{D} \in \mu$ -SDRL be a distribution and

$DRAW_{\mathcal{D}}$ an oracle that, when called, returns an element of Σ^* drawn according to \mathcal{D} . $DRAW_{\mathcal{D}}(n)$ will denote the multi-set obtained by n successive calls to $DRAW_{\mathcal{D}}$. Note that $DRAW_{\mathcal{D}}(n)$ produces a multi-set of strings that can be seen as a regular deterministic distribution.

$x^{-1}S$ will be the multi-set obtained from a multi-set S in which the prefix x has been deleted from the strings of S . More formally,

$$x^{-1}S = \{y \in \Sigma^* : z = xy \wedge z \in S\}_{multi}$$

Having the oracle $DRAW_{\mathcal{D}}(n)$ is not enough in general to efficiently identify or approximate a target regular distributions \mathcal{D} [1, 15]. More information related to the target concept must be given to the learning process. We now discuss the additional information one wants to provide to the learner.

Σ : Depending on the application, Σ can be given or not. In domain such as continuous speech recognition, an *a priori* vocabulary is given to the learning process. An *ad hoc* technique is used in order to deal with out of vocabulary words. In practice, the size of the vocabulary is a key factor for computational issues. Moreover, [1] showed that the problem of approximating an arbitrary distribution by non-deterministic regular distributions is exponential in the vocabulary size. In parallel, some techniques are used in practice to reduce the vocabulary size [12, 25].

Q_A : The set of states is not given. Nonetheless we give to our algorithm a bound on the size of Q_A . Even if the inference of automata of unbounded number of states has been studied [5, 16, 23, 26], from the practical point of view, a bound on the size is generally known. This bound either comes from domain experts or from computational constraints – large models will not fit in memory or will not be efficient.

δ_A : Describes the structure of the automaton. This function must be inferred from the data. Some authors limit the allowed structures of the model class to allow efficient inference [15, 22]. In the case of learning non deterministic distributions (*e.g.* estimation of Hidden Markov Model) the structure is usually given and the problem reduces to that of estimating the transition probability function.

P_A, F_A : When using maximum likelihood estimate and deterministic machines, these functions are totally defined as soon as δ_A is defined. Nonetheless, while working on real world data, some strings will not be parsed by the model either because they do not belong to the vocabulary¹, or because the model is not general enough. The model must then be smoothed: its probability functions are modified so that a non null probability is provided to each string of Σ^* . Although this problem has been studied from its theoretical aspects [18], it has mainly be considered from a practical point of view given its importance for the performance of the final application.

¹ A typical example is encountered in language modeling for speech recognition where proper names often provide out of vocabulary symbols.

Following [22], we will consider that the target concept will be μ -distinguishable for a given parameter μ . This implies that two states cannot be the same (up to a distance μ) in the model. This constraint can be seen as a constraint on the granularity of the model. Hence we are concerned with the inference of probabilistic deterministic μ -distinguishable finite state automata given a bound on the number of states (n), a finite alphabet Σ and an oracle *DRAW* that provides examples drawn according to the target concept.

4 The learning model

The learning model used here is motivated by the ones proposed in [15, 22].

Definition 3 (f-PAC). *Given a class of stochastic languages or distributions \mathcal{C} over Σ^* , an algorithm A f-Probably Approximately Correctly (f-PAC)-learns \mathcal{C} if there is a polynomial q such that for all c in \mathcal{C} , all $\epsilon > 0$ and $\delta > 0$, A is given a sample S_m drawn according to c , and produces a hypothesis H , such that $\Pr[f(c, H) > \epsilon] < \delta$ whenever $m > q(1/\epsilon, 1/\delta, |c|)$, where $|c|$ is some measure of the complexity of the class and $f(c, H) \geq 0$ measures a dissimilarity between c and H .*

In our particular case, the complexity of \mathcal{C} will be measured by a bound on the number of states of the elements of \mathcal{C} and by the distinguishability of the model class. The choice of f will be discussed later in the paper.

5 The ALISA algorithm

The classic strategy in grammatical inference consists in building a tree shape automaton – called a PPTA for Probabilistic Prefix Tree Acceptor – that is a representation of the maximum likelihood estimate of the data. Then, according to the particular algorithm, two states are chosen. If they are judged equivalent they are merged. The key point consists in deciding whether the states must be merged or not. In [5, 16, 22], the choice is made according to statistical tests applied to the distributions represented by the states (that is the distributions $A_{[q_0=q_j]}$ and $A_{[q_0=q_i]}$ are compared). In [26] a more global criterion is used, based on the analysis of the distribution modeled by the whole model before and after the merge. We follow here [22] and compare the distributions according to the distance L_∞ .

The main problems of the above strategy are the following: First it can lead to excessive space requirement. As an example, a PPTA built on the Wall Street Journal database has nearly one million states. As the size of databases naturally increases, and the size of the PPTA grows linearly with the size of the database, problems are inevitable. Secondly, when two states are considered equivalent, they are merged. The resulting state can then be used in other comparisons and other merges. This means that, on the one hand the independence of the statistical tests used is not satisfied and, on the other hand, that a wrong merge

will disturb following tests each time the given state will take part in a merge test. For these reasons, we will proceed incrementally: the solution will be built transition by transition².

Before the presentation of the algorithm, we provide an extension of the DPFA, the D-DPFA that are DPFA to which a new function that associate a distribution to each state is added.

Definition 4 (D-Dpfa). A D-DPFA is a tuple $\langle Q_A, \Sigma, \delta_A, q_0, F_A, P_A, S_A \rangle$ such that $Q_A, \Sigma, \delta_A, q_0, F_A, P_A$ define a DPFA and S_A is a function from Q_A to SDRL.

Algorithm 1 is the pseudo-code of the algorithm ALISA³ which works in two steps: First the structure is inferred and second the probability functions (transition and final) are estimated.

During the algorithm, the current solution will be a D-DPFA G . At each step of the algorithm (see figure 2 for support of the reading) we aim to create a new transition in G from a state of G (say ν_2 on figure 2). As an example, let us consider the transition $(\nu_2, \sigma, \nu_i) \notin \delta_G$. Let us note x the shorter prefix leading to ν_2 (i.e. $G \stackrel{x}{\rightsquigarrow} \nu_2$). The question now is to identify ν_i . We then consider the DPFA T built from a call to $DRAW_{\mathcal{D}}(N)$. In T , we find a candidate state cs such that $T \stackrel{x\sigma}{\rightsquigarrow} cs$. On figure 2 the state 2 is the candidate state cs . This state must match the end of the transition we aim to create (here ν_i). In order to find the matching state, we will compare the distribution $T_{[q_0=cs]}$ (represented by \mathcal{D}_{cs} on the figure) with each distribution associated to the states of G , that is the distributions $\mathcal{D}_{\nu_0}, \mathcal{D}_{\nu_1}, \mathcal{D}_{\nu_2}, \mathcal{D}_{\nu_3}$ on the figure⁴. If such a distribution exists, say the distribution associated with ν_3 , this means that $\nu_i = \nu_3$. We thus create the transition (ν_2, σ, ν_3) . If no state in G matches the candidate state cs , a new state is created in G and the distribution $(x\sigma)^{-1}DRAW_{\mathcal{D}}(N)$ associated to it.

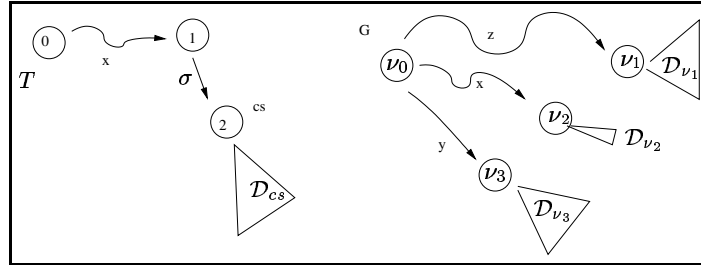


Fig. 2. Adding a new transition in G

² Our algorithm can be seen as a modification of the algorithm rlips [6] to guarantee the probable identification of the structure. We nevertheless propose a more algorithmic presentation of the algorithm.

³ ALISA stands for Algorithm for Learning Incrementally Stochastic Automata.

⁴ Note that another point of view would say that we compare the distributions $\mathcal{D}_{\nu_0}, \mathcal{D}_{\nu_1}, \mathcal{D}_{\nu_2}, \mathcal{D}_{\nu_3}$ with the distribution $(x\sigma)^{-1}DRAW_{\mathcal{D}}(N)$.

The process is repeated for each relevant⁵ state of T . When all the relevant states of T have been considered, there are three possibilities:

1. The user wants a solution: The transition and final probabilities are estimated and G is returned;
2. The structure of G is a complete graph or G has the required number of states according to the bound on the target number of states: The probabilities are estimated and G is returned;
3. Otherwise, a new sample is drawn and the process is repeated.

We now provide the formal definitions required for the description of the algorithm.

Definition 5 (σ -candidate-state/ exit state). A σ -candidate-state of a DPFA G according to a DPFA T , is a state q_i of T such that: $\exists x = \sigma_1 \dots \sigma_n$ and σ such that $T \stackrel{x}{\rightsquigarrow} q_i$, $G \stackrel{x}{\rightsquigarrow} \nu$, and $(\nu, \sigma, \bullet) \notin \delta_G$. The exit state is the target state of the path in G that recognizes x .

Note that an exit state is totally defined with the σ -candidate state. We will thus consider the exit state associated with a σ -candidate q , written $ES(q)$. On figure 2 state 2 of T is a σ -candidate state and state ν_2 is the exit state associated with the σ -candidate state 2: $ES(2) = \nu_2$.

Definition 6 defines what we mean by a *relevant state*:

Definition 6 (m_0 - σ -candidate state). A σ -candidate state q is a m_0 - σ -candidate state if its count is greater than m_0 : If $T \stackrel{x}{\rightsquigarrow} q$, then $|x^{-1}DRAW_{\mathcal{D}}(N)| \geq m_0$.

The comparison of the distributions $T_{[q_0=q_{cs}]}$ and $S(\nu_i)$, $\nu_i \in Q_G$ will be done using the following statistical result:

Lemma 1 ([3], lemma 4) Let \mathcal{D} be a distribution over Σ^* , and $a > 1$. For sufficiently large values of N , we have

$$\Pr \left[L_{\infty}(\mathcal{D}, DRAW_{\mathcal{D}}(N)) \leq \sqrt{6a(\log N)/N} \right] \geq 1 - 4N^{-a}.$$

In order to simplify notation, we will write $I(N) = \sqrt{6a(\log N)/N}$ and $\gamma(N) = 4N^{-a}$, $a > 1$.

6 Analysis of the algorithm

Given the distinguishability of the target μ and the confidence parameter δ , we need to define m_0 . More precisely, we need to define m_0 such that $I(m_0) \leq \mu/2$ – in order to distinguish μ -distinguishable states – and such that $\gamma(m_0) \leq \delta$ in order to have a sufficiently good confidence in the statistical decisions. The statistical test being applied on independent samples, we need to insure a global

⁵ The notion of relevance will be made more precise later in this section.

Algorithm 1: ALISA

Data : distinguishability: μ , Bound on the nb of states: n ,
statistical confidence: δ , alphabet: Σ

Result : A DPFA

Computation of N and m_0 /* see section 6 */

$G = \langle Q_G, \Sigma, \delta_G, \nu_0, P_G, F_G \rangle = \langle \{\nu_0\}, \Sigma, \emptyset, 0, 0, 0 \rangle$;

$S_G(\nu_0) \leftarrow \text{Build_PPTA}(\text{DRAW}_{\mathcal{D}}(N))$

while *Need to continue* **do**

$T \leftarrow \text{Build_PPTA}(\text{DRAW}_{\mathcal{D}}(N))$

foreach m_0 - σ -candidate state q_{cs} of Q_T **do**

 found $\leftarrow 0$

foreach state ν of Q_G **and** found = 0 **do**

1 **if** $L_{\infty}(T_{[q_0=q_{cs}]}, S_G(\nu)) \leq \mu$ **then**

2 | $\delta_G \leftarrow \delta_G \cup (ES(q_{cs}), \sigma, \nu)$

 | found $\leftarrow 1$

if found = 0 **then**

 /* no node were found similar: creation and link */

$Q_G \leftarrow Q_G \cup \{\nu'\}$

$S_G(\nu') \leftarrow T_{[q_0=q_{cs}]}$

$\delta_G \leftarrow \delta_G \cup (ES(q_{cs}), \sigma, \nu')$

 Estimate_Proba (G) /* see section 6.1 */

return G

confidence δ in order to satisfy the following constraints (with $a > 1$, and m_0 the threshold for a state to be relevant):

$$I(m_0) \leq \mu/2, \quad (1 - 4m_0^{-a}) \geq \delta \quad (4)$$

Note that m_0 is polynomial in $\frac{1}{\mu^2}$ since, as $\sqrt{n} \geq \ln(n)$ we have $\frac{\ln(n)}{n} \leq \frac{1}{\sqrt{n}}$ for $n \geq 1$ and thus $m_0 \geq (\frac{24a}{\mu^2})^2$ is enough to satisfy the first constraint.

If m_0 satisfies the above equations, and if N is greater than m_0 , then we are sure to produce, with a confidence greater than δ , an automaton isomorphic to the target:

Theorem 1 (G is isomorphic to the target) *If m_0 satisfies equations 4, then, at each time of the algorithm, each state ν of G matches a unique state q of the target C such that*

$$L_{\infty}(S_G(\nu), C_{[q_0=q]}) \leq \mu/2$$

Let us prove this by induction. At the initial step, the property is true since the unique state of G matches the right state (the initial one) and by lemma 1 the property is true.

Let us suppose now the theorem is true for each state of G and consider q_{cs} a m_0 - σ -candidate state of T and q its matching one in the target. Consider now two cases:

1. A state ν in G matches q . Then, by the triangular inequality
 $L_\infty(T_{[q_0=q_{cs}]}, S_G(\nu)) \leq L_\infty(T_{[q_0=q_{cs}]}, C_{[q_0=q]}) + L_\infty(C_{[q_0=q]}, S_G(\nu))$.
Thus, by lemma 1 and the definition of m_0 ,

$$L_\infty(T_{[q_0=q_{cs}]}, C_{[q_0=q]}) \leq \mu/2$$

By the inductive hypothesis and because m_0 respects equation 4,

$$\begin{aligned} L_\infty(C_{[q_0=q]}, S_G(\nu)) &\leq \mu/2 \\ \text{and thus } L_\infty(T_{[q_0=q_{cs}]}, S_G(\nu)) &\leq 2 \cdot (\mu/2) \end{aligned}$$

and the test of line 1 of the algorithm will return true.

2. No state ν of G matches q . Then no state in G will satisfy the line 1 of the algorithm. Indeed, let us consider q' in the target that matches state q_{cs} in T . Then, for each state ν of G , and q its matching state in C , we have:

$$\begin{aligned} L_\infty(C_{[q_0=q']}, C_{[q_0=q]}) &\leq L_\infty(C_{[q_0=q']}, T_{[q_0=q_{cs}]}) + \\ &\quad L_\infty(T_{[q_0=q_{cs}]}, S_G(\nu)) + L_\infty(S_G(\nu), C_{[q_0=q]}) \end{aligned}$$

by lemma 1 we have

$$L_\infty(C_{[q_0=q']}, T_{[q_0=q_{cs}]}) \leq \mu/2 \quad \text{and} \quad L_\infty(S_G(\nu), C_{[q_0=q]}) \leq \mu/2$$

By the distinguishability of the target, $L_\infty(C_{[q_0=q]}, C_{[q_0=q']}) > \mu/2$

and finally $L_\infty(T_{[q_0=q_{cs}]}, S_G(\nu)) > \mu$

and thus the test of the line 1 will return false for each state ν of G . A new state, say ν_n will be created and the algorithm will associate to it the distribution modeled by $T_{[q_0=q_{cs}]}$ which, by lemma 1 and because m_0 respects equation 4 satisfies $L_\infty(T_{[q_0=q_{cs}]}, C_{[q_0=q]}) \leq \mu/2$.

This shows that at each step of the algorithm the graph G is isomorphic to a subgraph of the target. When the algorithm stops, the automaton produced is isomorphic to the target, either because the target is a complete graph (and the algorithm will stop when the graph is complete) or because it has the required number of states: The automaton cannot have a number of states greater than the number of states of the target since it would require the creation of two states in the automaton matching the same state in the target. \square

We have just shown that the algorithm will produce with probability at least δ an automaton isomorphic to the target when given a polynomial number of example. Note here that if N is too small there is the possibility that no m_0 - σ -candidate state are available. In order to avoid this problem, we need to sample at least m_0/p with p the probability of having a string that goes through the state: $p = \min_{s \in Q_T} \max_{\pi \in T \xrightarrow{s}} \Pr_T(\pi)$. Adding p to the measure of the complexity of the model class is reasonable since learning automata with small p should be considered harder than learning ones with great p .

6.1 Estimating the probabilities

When a new transition (ν, σ, ν_i) is created in G , the probability of this transition can be estimated from the distribution $S_G(\nu)$ which, by construction, holds

enough information to provide a good estimate. This probability will be re-estimated later. Indeed, when no state is created, the end of the created transition is an existing state to which a distribution is associated. We can at that point add a new line (after the line numbered 2 in the algorithm) that will perform the merging of the distributions $T_{[q_0=q_{cs}]}$ and $S(\nu)$. The estimate of $S(\nu)$ will then be based on a greater amount of data and thus be more accurate.

6.2 f-PAC learnability issues

If we consider learning under the identification in the limit with probability one framework, we can use the technique of [9] which provides the identification in the limit of rational probabilities.

Let us now consider the more interesting f -PAC learning model where the measure of the complexity of the model class (written $|c|$ in the definition of the learning model) is expressed by $\frac{1}{\mu}$ and perhaps also by $\frac{1}{p}$.

When considering the choice of f , a link can be made with the non probabilistic case. Indeed, in the classic PAC framework, some negative results have been bypassed by restricting the class of the distributions from which the samples were drawn. This is true in particular with the regular languages that cannot be learned under the distribution free framework but where positive results exist when restrictions are provided on the distributions [20].

When considering the probabilistic case, the samples are drawn according to the target distribution. We are not yet in a distribution free framework. Nonetheless the difficulty of the learning can be expressed by the measure used to compare the inferred model and the target: When considering the distance L_∞ a simple maximum likelihood estimate is enough to guarantee the quality of the learning: According to Lemma 1, an algorithm that returns $DRAW_{\mathcal{D}}(N)$ will be L_∞ -PAC. Worse is the fact that an algorithm that just estimates the strings of probability greater than ε (according to the target) will learn under the L_∞ -PAC model. Thus, the L_∞ -PAC model is not a good one.

It is much harder to learn when the Kullback-Leibler divergence [8] is used as the quality measure. From the theoretical point of view, such a measure requires that the inferred model is close to the target for all strings, including the low probability ones. Even if this constraint is strong – in that an error on a low probability string can lead to a big value of the divergence – this is the one generally chosen while adapting the PAC framework to the probabilistic case [15, 18, 22]. From the practical point of view, the estimate of low probability strings is a key point in many applications such as speech recognition. The theoretical study of the smoothing is beyond the scope of this article. The interested reader can consider the theoretical works [18, 22], or more practical ones [11, 24, 17].

As a conclusion for this section, when considering n , Σ , μ , δ , and the oracle $DRAW_{\mathcal{D}}$, our algorithm provides an automaton isomorphic to the target with probability δ . When considering a theoretical guarantee of the quality of the final model, the choice of error function must be taken into account. If such a function is the L_∞ distance a simple estimation using the maximum likelihood estimate of the probabilities suffices to guarantee the quality of the model (using Chernoff

bounds). If the error is measured using the Kullback-Leibler divergence a deeper analysis is needed in order to take into account the low probability strings. From our opinion, the second framework is more appropriate given the practical importance of smoothing in real world applications. We have elsewhere shown [7] that such learning is possible.

7 Algorithm complexity

At each state of the algorithm (loop on m_0 - σ -candidate states) we consider adding a transition. This step is bounded by $n \times |\Sigma|$. For each of these candidate states, a state of G is considered. There are no more such state than the number of states in the target. The distance L_∞ is thus computed at most $n^2 \times |\Sigma|$. This computation is at most linear in the size of the sets $T_{[q_0=q_{cs}]}$ and $S_G(\nu)$, that is bounded by m_0 . The overall complexity of the algorithm is then $n^2 \times |\Sigma| \times m_0$. Note that according to its definition (equation 4) m_0 is polynomial in $1/\mu$ and $1/\delta$. Moreover, n is here the bound on the number of states of the *target* and not the size of the learning sample. The classic algorithms [5, 16, 26] are quadratic in the sample size. Here we are quadratic in the size of the *target* which is supposed to be much smaller.

8 Conclusion and further work

We have presented here a grammatical inference algorithm. This algorithm solves some of the classic problems of the most popular algorithms, since the independence of the statistical test is guaranteed and we have a clear sample complexity bound. Moreover, the algorithm is incremental (in that it does not proceed all the data at a time) and can be easily parallelized. From the theoretical point of view, we showed that this algorithm will infer, with high probability, an automaton isomorphic to the target using a polynomial amount of data. This result extends the one of [6] to a PAC like paradigm. The natural further work is the empirical comparison of the algorithm with other algorithms on natural data sets. Preliminary experiments on the Wall Street Journal database show that the ALISA algorithm is much faster in practice than some state merging algorithm [26].

References

1. N. Abe and W. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. In *Workshop on COLT*, pages 52–66, 1998.
2. P. Adriaans, H. Fernau, and M. van Zaannen, editors. *ICGI '02*, volume 2484 of *LNCS*, Berlin, Heidelberg, 2002. Springer-Verlag.
3. D. Angluin. Identifying languages from stochastic examples. Technical Report YALEU/DCS/RR-614, Yale University, Dept. of Computer Science, 1988.
4. Thorsten Brants. Estimating Markov model structures. In *ICSLP-96*, 1996.
5. R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *ICGI'94*, pages 139–152, 1994.

6. R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *Theoretical Informatics and Applications*, 33(1):1–20, 1999.
7. A. Clark and F. Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Jrl of Machine Learning Research*, 5:473–497, May 2004.
8. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Interscience Publication, 1991.
9. C de la Higuera and F. Thollard. Identification in the limit with probability one of stochastic deterministic finite automata. In de Oliveira [10].
10. A. de Oliveira, editor. *ICGI '00*, volume 1891 of *LNCS*, Berlin, Heidelberg, 2000. Springer-Verlag.
11. P. Dupont. Smoothing probabilistic automata: an error-correcting approach. In de Oliveira [10], pages 51–64.
12. P. Dupont and L. Chase. Using symbol clustering to improve probabilistic automaton inference. In *ICGI'98*, pages 232–243, 1998.
13. Y. Esposito, A. Lemay, F. Denis, and P. Dupont. Learning probabilistic residual finite state automata. In Adriaans et al. [2], pages 77–91.
14. D. Freitag. Using grammatical inference to improve precision in information extraction. In *Workshop on Grammatical Inference, Automata Induction, and Language Acquisition*, 1997.
15. M.J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.
16. C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In Adriaans et al. [2], pages 149–160.
17. D. Llorens, J. M. Vilar, and F. Casacuberta. Finite state language models smoothed using n-grams. *Int. Jrrl of Pattern Recognition and Artificial Intelligence*, 16(3):275–289, 2002.
18. D. McAllester and R. Shapire. On the convergence rate of the good-turing estimators. In *Thirteenth Annual Conf. on COLT*, pages 1–66, 2000.
19. M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. In *Workshop on Extended Finite-State Models of Language*, 1996.
20. R. Parekh and H. Honavar. Learning DFA from simple examples. In *International Colloquium on Machine Learning, ICML97*, 1997.
21. F. Pla, A. Molina, and N. Prieto. An Integrated Statistical Model for Tagging and Chunking Unrestricted Text. In P. Sojka, I. Kopeček, and K. Pala, editors, *Intl. Workshop on Text, Speech and Dialogue*, LNCS/LNAI 1902, pages 15–20, 2000.
22. D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *COLT 1995*, pages 31–40, USA, 1995. ACM.
23. A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, Dept. of Electrical Engineering and Computer Science, University of California at Berkeley, 1994.
24. F. Thollard. Improving probabilistic grammatical inference core algorithms with post-processing techniques. In *ICML*, pages 561–568, 2001.
25. F. Thollard and A. Clark. Shallow parsing using probabilistic grammatical inference. In Adriaans et al. [2], pages 269–282.
26. F. Thollard, P. Dupont, and C. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In Pat Langley, editor, *ICML*, 2000.
27. M. Young-Lai and F. WM. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning*, 40(2):111–137, 2000.